CSC 111: Intro to Computer Science through Programming

Spring 2017
Prof. Sara Mathieson
Homework 5 due tomorrow

Transcripts: only successful output at the end

Office hours today: 3-5pm in Ford 355

I will be away next week at a conference, but there will still be class, lab, and homework as usual
Outline: 3/20

- Mid-semester reminders
- Recap last time
- Continue Graphics
- Bullseye in Graphics
- This week: while loops and recursion
Mid-semester reminders
Syllabus reminders

- 10 hours outside of class every week
Syllabus reminders

+ 10 hours outside of class every week
+ Class attendance counts toward participation
Syllabus reminders

+ 10 hours outside of class every week
+ Class attendance counts toward participation
+ Piazza counts toward participation (asking and answering)
Syllabus reminders

- 10 hours outside of class every week
- Class attendance counts toward participation
- Piazza counts toward participation (asking and answering)
- Lab attendance is required (and must attend your own section)
Syllabus reminders

- 10 hours outside of class every week
- Class attendance counts toward participation
- Piazza counts toward participation (asking and answering)
- Lab attendance is required (and must attend your own section)
- Each partner must submit the lab on Moodle
Syllabus reminders

+ 10 hours outside of class every week
+ Class attendance counts toward participation
+ Piazza counts toward participation (asking and answering)
+ Lab attendance is required (and must attend your own section)
+ Each partner must submit the lab on Moodle
+ Homeworks: cite your collaborators at the top (comment)
Syllabus reminders

+ 10 hours outside of class every week
+ Class attendance counts toward participation
+ Piazza counts toward participation (asking and answering)
+ Lab attendance is required (and must attend your own section)
+ Each partner must submit the lab on Moodle
+ Homeworks: cite your collaborators at the top (comment)
+ Homeworks: individual, original code, produced and understood by you
10 hours outside of class every week

Class attendance counts toward participation

Piazza counts toward participation (asking and answering)

Lab attendance is required (and must attend your own section)

Each partner must submit the lab on Moodle

Homeworks: cite your collaborators at the top (comment)

Homeworks: individual, original code, produced and understood by you

No late work without an accommodations letter or deans email
Recap
(+ cat examples)
Christine
Ruth
Zoe
Informal quiz (discuss with a partner)

1) If I have \( p = \text{Point}(40,60) \) and then say \( p.x = 80 \), \( x \) here is called an ________________

2) What is wrong with the following code to draw two eyes?

```python
left_eye = \text{Circle}(\text{Point}(225,275),25)
left_eye.setFill("light blue")
left_eye.draw(win)

right_eye = left_eye
right_eye.move(150,0)
right_eye.draw(win)
```

3) \text{Circle}(..) and \text{Point}(..) are __________

4) \text{setFill}(..), move(..), draw(..) are __________ not __________

5) \text{left_eye} and \text{right_eye} are ______________ of the Circle ________

6) Extra: are graphics objects mutable or immutable?
Informal quiz (discuss with a partner)

1) If I have $p = \text{Point}(40,60)$ and then say $p.x = 80$, $x$ here is called an *instance variable*.

2) What is wrong with the following code to draw two eyes?

```java
left_eye = Circle(Point(225,275),25)
left_eye.setFill("light blue")
left_eye.draw(win)

right_eye = left_eye
right_eye.move(150,0)
right_eye.draw(win)
```

*Need to use .clone()*

3) $\text{Circle(..)}$ and $\text{Point(..)}$ are *constructors*

4) $\text{setFill(..)}, \text{move(..)}, \text{draw(..)}$ are *methods* not *functions*

5) $\text{left_eye}$ and $\text{right_eye}$ are *instances* of the Circle *class*

6) Extra: are graphics objects mutable or immutable? *Mutable*
Continue: OOP
Instance Variables

We have already seen example of methods, which belong to a specific instance of a class. Example: `<Point instance>.draw(…)

```python
>>> p = Point(50, 75)
>>> p.draw(win)
```
Instance Variables

- We have already seen example of methods, which belong to a specific instance of a class. Example: <Point instance>.draw(…)

```python
>>> p = Point(50,75)
>>> p.draw(win)
```

- Instances also have *instance variables*, which can be accessed and/or modified. We also use a “dot”, but no parentheses!

```python
>>> p.x
50.0
>>> p.y
75.0
```
GraphWin class

+ **GraphWin**(title, width, height) – constructs a new graphics window (default width and height are both 200)

+ **setBackground**(color) – set the background color

+ **close**() – closes the window

+ **getMouse**() – waits for the user to click, returns the click position as a **Point**
Methods for all Graphics Objects

- `setFill(color)` – sets the interior color of an object
- `setOutline(color)` – sets the outline color of an object
- `setWidth(pixels)` – sets the outline width (doesn’t work for `Point`)
- `draw(window)` – draws the object on the given window
- `undraw()` – removes the object from a graphics window
- `move(dx, dy)` – moves the object dx in the x direction and dy in the y direction
- `clone()` – returns a duplicate (new copy) of the object
Point class

+ **Point(x,y)** – constructs a new point at the given position
+ **getX()** – returns the current x coordinate
+ **getY()** – returns the current y coordinate
Line class

- **Line(point1, point2)** – constructs a line from point1 to point2
- **setArrow(string)** – sets the arrowhead of a line ("first", "last", "both", "none")
- **getCenter()** – returns the midpoint of the line
- **getP1(), getP2()** – returns a clone of the corresponding endpoint

Zelle Chap 4: page 110
Circle class

+ **Circle(center, radius)** – constructs a circle at the given position and with the given radius

+ **getCenter()** – returns a clone of the center point

+ **getRadius()** – returns the radius

+ **getP1(), getP2()** – returns a clone of the corresponding corner of the circle’s bounding box (upper left, lower right)

Zelle Chap 4: page 110
Rectangle class

+ **Rectangle(point1, point2)** – constructs a rectangle with opposite corners at the given points (upper left, lower right)

+ **getCenter()** – returns the center point

+ **getP1(), getP2()** – returns a clone of the corner point
Polygon class

- `Polygon(point1, point2, point3, ...)` – constructs a polygon with the given points as vertices (also accepts a list of points)
- `getPoints()` – returns a list of the points in the polygon
Text class

+ **Text(point, content)** – constructs a text object with at the given point and with the string content

+ **setTextColor(color)** – set text color

+ **setSize(size)** – set the text size (i.e. size = 14)
Bullseye Exercise
Step 1: create 10 concentric circles

- Window 450 x 450
- Largest circle has radius 200
- Radii decrease by 20 each time
Step 2: color the circles using a list

- Set up a list of colors at the top of your code
- I would recommend having your list be of length 10
- Index into this list to color each circle
Step 3: create text for a score

- Text(p, s) where p is a Point and s is a string
- Methods:
  - setTextColor(color)
  - setSize(size)

http://wiki.tcl.tk/37701